NORDUGRID-MANUAL-13
27/3/2015

ARC Clients

User Manual for ARC 11.05 (client versions 1.0.0) and above

Contents

1 Introduction **5**

2 Commands **7**

2.1 Proxy utilities . . . 7

2.1.1 arcproxy . . . 7

2.2 Job submission and management . . . 11

2.2.1 arcsub . . . 11

2.2.2 arcstat . . . 16

2.2.3 arccat . . . 17

2.2.4 arcget . . . 18

2.2.5 arcsync . . . 19

2.2.6 arcinfo . . . 20

2.2.7 arckill . . . 21

2.2.8 arcclean . . . 22

2.2.9 arcrenew . . . 24

2.2.10 arcresume . . . 25

2.2.11 arcresub . . . 26

2.3 Data manipulation . . . 28

2.3.1 arcls . . . 28

2.3.2 arccp . . . 29

2.3.3 arcrm . . . 30

2.3.4 arcmkdir . . . 31

2.3.5 arcrename . . . 31

2.4 Test suite . . . 32

2.4.1 arctest . . . 32

3 URLs **35**

4 ARC Client Configuration **39**

4.1 Block `[common]` . . . 40

defaultservices . . . 40

rejectservices . . . 40

rejectdiscovery . . . 40

rejectmanagement . . . 41

infointerface . . . 41
submissioninterface . . . 41
verbosity . . . 41
timeout . . . 41
brokername . . . 41
brokerarguments . . . 42
joblist . . . 42
joblisttype . . . 42
bartender . . . 42
proxypath . . . 42
keypath . . . 43
certificatepath . . . 43
cacertificatesdirectory . . . 43
cacertificatepath . . . 43
vomsserverpath . . . 43
jobdownloaddirectory . . . 43
4.2 Service blocks . . . 44
url . . . 44
default . . . 44
group . . . 44
infointerface . . . 45
submissioninterface . . . 45
registryinterface . . . 45
4.3 srms.conf . . . 46
4.4 Block `[alias]` . . . 46
4.5 Deprecated configuration files . . . 46

Chapter 1

Introduction

The command line user interface of ARC consists of a set of commands necessary for job submission and manipulation and data management. This manual replaces the older version in `NORDUGRID-MANUAL-1` and is valid for ARC release 11.05 (client versions 1.0.0) and above. Command line tools semantics are the same as in earlier (0.x) versions of ARC, roughly following that of basic Linux commands and most common batch system commands. One obvious difference is change of the legacy prefix from "ng" to the more appropriate "arc". This is not only a cosmetic change: **behaviour of the commands also have changed**, as did their functionalities and options.

> Users are strongly discouraged from modifying their old scripts by simply replacing "ng" with "arc" – results may be unpredictable.

Chapter 2

Commands

2.1 Proxy utilities

ARC now comes complete with a set of utilities to create temporary user credentials (proxies) used to access Grid services.

2.1.1 arcproxy

In order to contact Grid services (submit jobs, copy data, check information etc), one has to present valid credentials. These are commonly formalized as so-called "proxy" certificates. There are many different types of proxy certificates, with different Grids and different services having own preferences. `arcproxy` is a powerful tool that can be used to generate most commonly used proxies. It supports the following types:

- pre-RFC GSI proxy
- RFC-compliant proxy (default)
- VOMS-extended proxy
- MyProxy delegation

`arcproxy` requires presence of user's private key and public certificate, as well as the public certificate of their issuer CA. These certificates can exist either as separate files, or in an NSS certificate store, such as that used by Firefox. That is, if your certificate is in your Firefox browser, there is no need to export it into files, as `arcproxy -F` can be used to create a proxy directly.

arcproxy [options]

Options:

`-P, --proxy`	*path*	path to the generated proxy file, defaults are described below
`-C, --cert`	*path*	path to the certificate file, defaults are described below
`-K, --key`	*path*	path to the key file, defaults are described below
`-T, --cadir`	*path*	path to the trusted certificate directory, only needed for VOMS client functionality; defaults are described below

`-s, --vomsdir`	*path*	path to the top directory of VOMS *.lsc files, only needed for the VOMS client functionality
`-V, --vomses`	*path*	path to the VOMS server configuration file; if the path is a directory rather than a file, all the files under this directory will be searched
`-S, --voms`	*voms[:command]*	Specify VOMS server (more than one VOMS server can be specified like this: –voms VOa:command1 –voms VOb:command2) :command is optional, and is used to ask for specific attributes(e.g. roles). Command options are: all – put all of this DN's attributes into AC; list – list all of the DN's attribute,will not create AC extension; /Role=yourRole – specify the role, if this DN has such a role, the role will be put into AC /voname/groupname/Role=yourRole – specify the VO,group and role; if this DN has such a role, the role will be put into AC
`-o, --order`	*group[:role]*	Specify ordering of attributes, examples: -o /knowarc.eu/coredev:Developer,/knowarc.eu/testers:Tester
`-G, --gsicom`		use GSI communication protocol for contacting VOMS services
`-H, --httpcom`		use HTTP communication protocol for contacting VOMS services that provide RESTful access. Note that for the RESTful access, 'list' command and multiple VOMS servers are not supported.
`-O, --old`		use GSI proxy (default is RFC 3820 compliant proxy)
`-I, --info`		print all information about this proxy; in order to show the Identity (DN without CN as suffix for proxy) of the certificate, the trusted certificates directory (`-T, --cadir`) is needed
`-r, --remove`		removes the proxy file
`-U, --user`	*string*	username for the MyProxy server
`-N, --nopassphrase`		don't prompt for a credential passphrase when retrieving a credential from on MyProxy server The precondition of this choice is that the credential is PUT onto the MyProxy server without a passphrase by using `-R (--retrievable_by_cert)` option when being PUTing onto MyProxy server. This option is specific for the GET command when contacting a MyProxy server.
`-R, --retrievable_by_cert`	*string*	Allow specified entity to retrieve credential without passphrase; this option is specific for the PUT command when contacting MyProxy server.
`-L, --myproxysrv`	*URL*	URL of MyProxy server, optionally followed by colon and port number, e.g. `example.org:7512`; if the port number is not specified, 7512 is used by default
`-M, --myproxycmd`	*PUT|GET*	command to MyProxy server; the command can be PUT or GET: PUT/put – put a delegated credential to MyProxy server;

		GET/get – get a delegated credential from MyProxy server, credential (certificate and key) is not needed in this case
`-F, --nssdb`		use NSS credential DB in default Mozilla profiles, including Firefox, Seamonkey and Thunderbird
`-c, --constraint`	*string*	proxy constraints (see options below)
`-t, --timeout`	*seconds*	timeout for network communication, in seconds (default 20 seconds)
`-d, --debug`	*verbosity*	verbosity level is one of FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG
`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page

MyProxy functionality can be used together with VOMS functionality, i.e., a credential stored in a MyProxy server can receive a VOMS AC.

If destination location of proxy file is not specified with option `-P`, the value of `X509_USER_PROXY` environment variable is used explicitly. If no such value is provided, the default location is used: `<TEMPORARY DIRECTORY>/x509up_u<USER ID>`. Here `TEMPORARY DIRECTORY` is derived from environment variables `TMPDIR`, `TMP`, `TEMP` or default location `/tmp` is used.

The (public) certificate file as specified with option `-C` can be either *pem*, *der*, or *pkcs12* formated. If this option is not set, then the path specified by the environment variable `X509_USER_CERT` will be searched. If `X509_USER_CERT` is not set, then the `certificatepath` attribute in the client configuration file (`client.conf`) will be used. If after all these attempts the certificate still is not found, then file `usercert.pem` wll be searched in `/.arc/`, `/.globus/`, `./etc/arc`, and `./`.

If the certificate file as specified with option `-C` is in *pkcs12* format, then no need to specify private key with option `-K`. If option `-K` is not set and the certificate is not *pkcs12*, then the path specified by the environment variable `X509_USER_KEY` will be searched. If `X509_USER_KEY` is not set, then the *keypath* attribute in the client configuration file (`client.conf`) will be used. If after all these attempts the key still is not found, then file `userkey.pem` will be searched in `/.arc/`, `/.globus/`, `./etc/arc`, and `./`.

If option `-T` is not set, then the path specified by the environment variable `X509_CERT_DIR` will be searched. If `X509_CERT_DIR` is not set, then the *cacertificatesdirectory* attribute in the client configuration file (`client.conf`) will be used.

The `-o, --order` attribute can be used several times, e.g.:

```
--order /knowarc.eu/coredev:Developer --order /knowarc.eu/testers:Tester
```

Note that it does not make sense to specify the order if you have two or more different VOMS server specified.

When getting the delegated credentials from a MyProxy server using the `-M` option, regular credentials (certificate and key) are not needed. MyProxy functionality can be used together with VOMS functionality. Options `--voms` and `--vomses` can be used for the `GET` command if VOMS attributes are required to be included in the proxy.

Supported constraints to be specified with the option `-c` are:

- `validityStart=time` – e.g. 2008-05-29T10:20:30Z; time when certificate becomes valid. Default is now.
- `validityEnd=time` – time when certificate becomes invalid. Default is 43200 (12 hours) from start.
- `validityPeriod=time` – e.g. 43200 or 12h or 12H; for how long certificate is valid. If neither `validityPeriod` nor `validityEnd` are specified, default is 12 hours for local proxy, and 168 hours for delegated proxy on MyProxy server.

- `vomsACvalidityPeriod=time` – e.g. 43200 or 12h or 12H; for how long the VOMS AC is valid. Default is the least value of 12 hours and `validityPeriod`.
- `myproxyvalidityPeriod=time` – duration of proxies delegated by MyProxy server, e.g. 43200 or 12h or 12H; if not specified, the default is the least value of 12 hours and `validityPeriod` (which is the life time of the delegated proxy on a MyProxy server).
- `proxyPolicy=policy content` – assigns specified string to proxy prolicy to limit its functionality.
- `proxyPolicyFile=policy file`
- `keybits=number` – length of the key to generate. Default is 1024 bits. Special value *inherit* means using the same key length as the signing certificate.
- `signingAlgorithm=name` – signing algorithm to use for signing the public key of the proxy. Possible values are *sha1*, *sha2* (alias for *sha256*), *sha224*, *sha256*, *sha384*, *sha512* and *inherit* (use algorithm of the signing certificate). Default is *inherit*.

Proxy information items requested with the `-i` option are printed in the requested order and are separated by newline. If an item has multiple values, they are printed in same line separated by a pipe sign (|). Supported information item names are:

- `subject` – subject name of the proxy certificate.
- `identity` – identity subject name of the proxy certificate.
- `issuer` – issuer subject name of the proxy certificate.
- `ca` – subject name of the CA which issued the initial certificate.
- `path` – file system path to the file containing proxy.
- `type` – type of the proxy certificate.
- `validityStart` – timestamp when proxy validity starts.
- `validityEnd` – timestamp when proxy validity ends.
- `validityPeriod` – duration of proxy validity in seconds.
- `validityLeft` – duration of remaining proxy validity in seconds.
- `vomsVO` – VO name represented by the VOMS attribute.
- `vomsSubject` – subject of the certificate for which the VOMS attribute was issued.
- `vomsIssuer` – subject of the service which issued the VOMS certificate.
- `vomsACvalidityStart` – timestamp when the VOMS attribute validity starts.
- `vomsACvalidityEnd` – timestamp when the VOMS attribute validity ends.
- `vomsACvalidityPeriod` – duration of the VOMS attribute validity in seconds.
- `vomsACvalidityLeft` – duration of the remaining VOMS attribute validity in seconds.
- `proxyPolicy` – policy associated with the proxy.
- `keybits` – size of the proxy certificate key in bits.
- `signingAlgorithm` – algorithm used to sign the proxy certificate.

`arcproxy` makes use of the following configuration files:

`/etc/vomses`: common for all users file containing a list of selected VO contact points, one VO per line, for example:

```
"gin" "kuiken.nikhef.nl" "15050" "/O=dutchgrid/O=hosts/OU=nikhef.nl/CN=kuiken.nikhef.nl" "gin.ggf.org"
"nordugrid.org" "voms.uninett.no" "15015" "/O=Grid/O=NorduGrid/CN=host/voms.ndgf.org" "nordugrid.org"
```

`~/.voms/vomses`: same as `/etc/vomses` but located in user's home area. If exists, has precedence over `/etc/vomses`. The order of the parsing of `vomses` location is:

1. command line options
2. client configuration file `~/.arc/client.conf`
3. `$X509_VOMSES` or `$X509_VOMS_FILE`
4. `~/.arc/vomses`
5. `~/.voms/vomses`
6. `$ARC_LOCATION/etc/vomses` (for Windows environment)
7. `$ARC_LOCATION/etc/grid-security/vomses` (for Windows environment)
8. `$PWD/vomses`
9. `/etc/vomses`
10. `/etc/grid-security/vomses`

`~/.arc/client.conf`: the overall ARC client configuration file, see Section 4. Some options can be given default values by specifying them in such ARC client configuration file. By using the `--conffile` option a different configuration file can be used rather than the default.

2.2 Job submission and management

The following commands are used for job submission and management, such as status check, results retrieval, cancellation, re-submission and such. The jobs must be described using a job description language. ARC supports the following languages: JSDL [1], xRSL [7] and JDL [5].

2.2.1 arcsub

The `arcsub` command is the most essential one, as it is used for submitting jobs to the Grid resources. `arcsub` matches user's job description to the information collected from the Grid, and the optimal site is being selected for job submission. The job description is then being submitted to that site, and usually is then forwarded to a local Resource Management System (LRMS), which can be, e.g., PBS or Condor or SGE etc.

arcsub [options] [filename ...]

Options:

`-e, --jobdescrstring`	*filename*	string describing the job to be submitted
`-f, --jobdescrfile`	*filename*	file describing the job to be submitted
`-j, --joblist`	*filename*	the file storing information about active jobs (on Linux, default $/.arc/jobs.dat)
`-o, --jobids-to-file`	*filename*	the IDs of the submitted jobs will be appended to this file
`-D, --dryrun`		add dryrun option to the job description
`-x, --dumpdescription`		do not submit – dump transformed job description to stdout
`-b, --broker`	*string*	select broker method (default is Random)

`-P, --listplugins`		list the available plugins
`-t, --timeout`	*seconds*	timeout for network communication, in seconds (default 20)
`-d, --debug`	*verbosity*	verbosity level, FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG - default WARNING
`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page
Options in ARC 11.05:		
`-c, --cluster`	`[-]`*designator*	explicitly select or reject (-) a specific site
`-g, --index`	`[-]`*designator*	explicitly select or reject (-) a specific index server
Options in ARC 12.05:		
`-c, --cluster`	*designator*	select one or more computing elements by an alias for a single CE, a group of CEs or a URL
`-g, --index`	*designator*	select one or more registries by an alias for a single registry, a group of registries or a URL
`-R, --rejectdiscovery`	*URL*	skip the service with the given URL during service discovery
`-S, --submissioninterface`	*InterfaceName*	only use this interface for submitting (e.g. org.nordugrid.gridftpjob, org.ogf.glue.emies.activitycreation, org.ogf.bes)
Options in ARC 13.02:		
`--direct`		submit directly - no resource discovery or matchmaking
`-I, --infointerface`	*InterfaceName*	the computing element specified by URL at the command line should be queried using this information interface (possible options: org.nordugrid.ldapng, org.nordugrid.ldapglue2, org.nordugrid.wsrfglue2, org.ogf.glue.emies.resourceinfo)
Arguments:		
`filename ...`		file(s) describing the job(s) to be submitted

A typical Grid job submission looks like:

```
arcsub myjob.jsdl
```

Here `myjob.jsdl` is a file containing job description. Note that in this example `-f` is omitted since the job description file is the last item on the command line.

Please remember that you must have **valid credentials** (see Section 2.1) and be authorised at at least one Grid site.

The job must be described using one of the supported job description languages. The description can be entered either as an argument on the command line, or can be read from a file, as in the example above. Several jobs can be requested at the same time by giving more than one file name as an argument, or by repeating the `-f` or `-e` options. It is possible to mix `-e` and `-f` options in the same `arcsub` command.

A simple *"Hello World"* job description `myjob.jsdl` using the standard JSDL language is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<JobDefinition
 xmlns="http://schemas.ggf.org/jsdl/2005/11/jsdl"
 xmlns:posix="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix">
```

```
 <JobDescription>
   <JobIdentification>
     <JobName>Hello World job</JobName>
   </JobIdentification>
   <Application>
     <posix:POSIXApplication>
       <posix:Executable>/bin/echo</posix:Executable>
       <posix:Argument>'Hello World'</posix:Argument>
       <posix:Output>out.txt</posix:Output>
       <posix:Error>err.txt</posix:Error>
     </posix:POSIXApplication>
   </Application>
 </JobDescription>
</JobDefinition>
```

If a job is successfully submitted, a **job identifier** (*job ID*) is printed to standard output.

The job ID uniquely identifies the job. Job IDs differ strongly for different computing service flavours, but basically they have a form of a URL. You should use job ID as a handle to refer to the job when doing other job manipulations, such as querying job status (`arcstat`), killing it (`arckill`), re-submitting (`arcresub`), or retrieving the result (`arcget`).

Usually job ID is a valid URL for the job session directory. You can almost always use it to access the files related to the job, by using data management tools (see Chapter 2.3).

There may be exceptions for some computing service flavours like CREAM which do not support listing job session directory.

The `-c` option can be used to manually select known computing sites, for example:

```
arcsub -c alias -c group -c URL job.xrsl
```

This will submit a job to either of the sites known by the alias `alias` or member of the group `group`* or having the URL `URL`.

In ARC 11.05, to submit a job to any site except `badsite`, use `-` sign in front of the name:

```
arcsub -c -badsite job.xrsl
```

See below for description of different kinds of designators which can be used with `-c` and `-g` options in ARC 11.05.

In ARC 12.05 and higher, to submit a job to any site except `badsite.example.com`, use the `-R` option with the site's URL (which can be shortened to a domain name):

```
arcsub -R badsite.example.com job.xrsl
```

The `arcsub` command locates the available sites by querying the information system (unless option `-c` is used, in which case only the listed sites are queried). Default index services for the information system are specified in the configuration template distributed with the middleware, and can be overwritten both in the user's configuration (see Section 4) and from the command line using option `-g`. Different Grid middlewares may use different notation for such index services.

In ARC 11.05 (client versions 1.*.*) the designators for `-c` and `-g` are either alias names, group names or URLs. `interface:URL`, where `interface:` is optional, specifying the computing service flavour (and the corresponding plugin) to be used when handling the URL. Possible flavours are:

*groups are only available in ARC 12.05

`ARC0` Legacy ARC execution and index services (requires the `nordugrid-arc-plugins-globus` package to be installed)

`ARC1` Web service ARC execution service derived from OGSA-BES standard

`CREAM` CREAM BES-compliant execution service

`BES` Generic BES plugin consistent with the OGSA-BES standard

`EMIES` Web service following EMI Execution Service specifications

Here are examples of full designators for ARC legacy index services:

```
ARC0:ldap://ce.ng.eu:2135/nordugrid-cluster-name=ce.ng.eu,Mds-Vo-name=local, o=grid
ARC0:ldap://index.ng.org:2135/mds-vo-name=sweden,O=grid
```

and for CREAM

```
CREAM:ldap://cream.glite.org:2170/o=grid
```

In case *interface:* part is missing every communication protocol/interface corresponding to supported flavours and matching URL will be tried. Because `arcsub` supports multiple Grid flavours and this number is continuously increasing it is strongly advisable not to skip *interface:* part. Example of such designator is

```
ldap://ce.ng.eu:2135/nordugrid-cluster-name=ce.ng.eu,Mds-Vo-name=local,o=grid
```

For convenience it is possible to shorten designator even more by skipping protocol and path parts of URL. So designator may be as simple as hostname of service to be contacted. Here is an example of such shorthand designator for index server

```
index.ng.org/mds-vo-name=sweden
```

and those suitable both for `-c` and `-g` options:

```
cream.glite.org
ce.ng.eu
```

If such short designators are used then rest of the URL is automatically generated according to the flavour which is currently tried. For example in the case of ARC1, `https` communication protocol is assumed.

If you are using some services frequently enough it is recommended to use aliases for these URLs. Aliases are specified in the configuration file (see Section 4).

In ARC 12.05 and higher, it is advisable to configure each site in your client configuration (see Section 4.2) and use its alias. Specifying designators (full URLs) on the command line is possible, but not recommended. Flavours as used in ARC 11.05 are not recognised in ARC 12.05. To list possible submission interfaces (to use with `-S` option), use `arcinfo` command.

In order to keep track of submitted jobs, ARC client stores information in a dedicated file, on Linux platforms by default located in `$HOME/.arc/jobs.dat`. It is sometimes convenient to keep separate lists (e.g., for different kinds of jobs), to be used later with e.g. `arcstat`. This is achieved with the help of the `-j` command line option.

The ARC client transforms input job description into a format that can be understood by the Grid services to which it is being submitted. By specifying the `--dumpdescription` option, such transformed description is written to the standard output instead of being submitted for execution.

Possible broker values for the `arcsub` command line option `-b` are:

– `Random` – ranks targets randomly (default)

- `FastestQueue` – ranks targets according to their queue length
- `Benchmark[:name]` – ranks targets according to a given benchmark, as specified by the `name`. If no benchmark is specified, CINT2000 † is used
- `Data` – ranks targets according the amount of megabytes of the requested input files that are already in the computing resources cache. Note that only targets running the A-REX BES interface can supply this information.
- `PythonBroker:<module>.<class>[:arguments]` – ranks targets using any user-supplied custom Python broker module, optionally with broker arguments. Such module can reside anywhere in user's `PYTHONPATH`
- `<otherbroker>[:arguments]` – ranks targets using any user-supplied custom C++ broker plugin, optionally with broker arguments. Default location for broker plugins on Linux systems is `/usr/lib/arc` (may depend on the operating system), or the one specified by the `ARC_PLUGIN_PATH`.

To write a custom broker in C++ one has to write a new specialization of the `BrokerPlugin` base class and implement the methods `set()`, `match()` and `operator()` in the new class. The class should be compiled as a loadable module that has the proper ARC plugin descriptor for the new broker. For example, to build a broker plugin "MyBroker" one executes:

```
g++ -I /arc-install/include \
    -L /arc-install/lib \
    ‘pkg-config --cflags glibmm-2.4 libxml-2.0‘ \
    -o libaccmybroker.so -shared MyBroker.cpp
```

For more details, refer to *libarclib* documentation [2].

It often happens that some sites that `arcsub` has to contact are slow to answer, or are down altogether. This will not prevent you from submitting a job, but will slow down the submission. To speed it up, you may want to specify a shorter timeout (default is 20 seconds) with the `-t` option:

```
arcsub -t 5 myjob.jsdl
```

Default value for the timeout can be set in the user's configuration file (see Section 4).

If you would like to get diagnostics of the process of resource discovery and requirements matching, a very useful option is `-d`. The following command:

```
arcsub -d VERBOSE myjob.xrsl
```

will print out the steps taken by the ARC client to find the best cluster satisfying your job requirements. Possible diagnostics levels, in the order of increasing verbosity, are: `FATAL`, `ERROR`, `WARNING`, `INFO`, `VERBOSE` and `DEBUG`. Default is `WARNING`, and it can be set to another value in the user's configuration file.

Default configuration file on Linux platforms is `$HOME/.arc/client.conf`. However, a user can choose any other pre-defined configuration through option `-z`.

Command line option `-v` prints out version of the installed ARC client package, and option `-h` provides a short help text.

For certain advanced computational jobs which may need to communicate their status to some external services, there may be a need for knowing internal job ID. For jobs accepted by ARC computational services this information can be found in the local (for the job executable) environment variable `GRID_GLOBAL_JOBID`. One needs to take into account that this ID is probably different from the one provided by `arcsub`. An example is an ID provided by the A-REX computing service. That service provides OGSA-BES compatible interface for job management and the ID contains an XML document compliant with OGSA-BES specifications.

†http://www.spec.org/cpu2000/CINT2000/

2.2.2 arcstat

arcstat [options] [job ...]

Options:

`-a, --all`		all jobs
`-j, --joblist`	*filename*	the file storing information about active jobs (on Linux, default $HOME/.arc/jobs.dat)
`-i, --jobids-from-file`	*filename*	file containing a list of job IDs
`-s, --status`	*statusstr*	only select jobs whose status is *statusstr*
`-l, --long`		long format (extended information)
`-S, --sort`	*criterion*	sort jobs according to job ID (criterion *jobid*), submission time (*submissiontime*) or job name (*jobname*)
`-R, --rsort`	*criterion*	reverse sorting of jobs according to job ID, submission time or job name
`-u, --show-unavailable`		show jobs where status information is unavailable
`-p, --print-jobids`		instead of the status only the IDs of the selected jobs will be printed
`-P, --listplugins`		list the available plugins
`-t, --timeout`	*time*	timeout for queries (default 20 sec)
`-d, --debug`	*verbosity*	verbosity level is one of FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG
`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page
Options in ARC 11.05:		
`-c, --cluster`	`[-]`*name*	explicitly select or reject a specific site
Options in ARC 12.05:		
`-c, --cluster`	*designator*	select one or more computing elements by an alias for a single CE, a group of CEs or a URL
`-r, --rejectmanagement`	*URL*	skip jobs which are on a computing element with a given URL

Arguments:

`job ...`	list of job IDs and/or job names

The `arcstat` command returns the status of jobs in the Grid, and is typically issued with a job ID (as returned by `arcsub`) as an argument. It is also possible to use job name instead of ID, but if several jobs have identical names, information will be collected about all of them. More than one job ID and/or name can be given.

When several of the `-c <cluster>`, `-i <jobidfile>` and `[job...]` command line options are specified, the command returns information about all jobs listed on the command line plus all jobs on the specified clusters plus all jobs from the specified `jobidfile`.

However the `-c -<cluster>` (for ARC 11.05), `-r <URL>` (for ARC 12.05) and `-s <status>` options will filter the jobs selected by the above mentioned options, or if none of those are specified, then

these will filter all the jobs.

For example, `arcstat -s Finished -c mycluster <jobid>` will return information about the finished jobs on `mycluster` plus about `<jobid>` but only if it is finished. Or, `arcstat -i jobidfile -r mycluster.example.com` will return information about jobs which are in the `jobidfile` but not on `mycluster.example.com`.

If the `-l` option is given, extended information is printed.

Jobs can be sorted according to the job ID, submission time or job name, either in normal or reverse order. By using the `--sort` or `--rsort` option followed by the desired ordering (`jobid`, `submissiontime` or `jobname`, respectively), jobs will be sorted in normal or reverse order. Note that the options `--sort` and `--rsort` cannot be used at the same time.

Options `-a`, `-c`, `-s` and `-j` do not use job ID or names. By specifying the `-a` option, the status of all active jobs will be shown. If the `-j` option is used, the list of jobs is read from a file with the specified filename, instead of the default one (`$HOME/.arc/jobs.dat`) (on Linux).

Option `-c` accepts arguments as explained in the description of `arcsub`, that is, in the `GRID:URL` notation for ARC 11.05, or URLs, aliases and groups from the configuration file in ARC 12.05.

Different sites may report different job states, depending on the installed grid middleware version. Typical values can be e.g. “Accepted”, “Preparing”, “Running”, “Finished” or “Deleted”. Please refer to the respective middleware documentation for job state model description.

Command line option `-s` will instruct the client to display information of only those jobs which status matches the instruction. This option must be given together with either `-a` or `-c` ones, e.g.:

```
arcstat -as Finished
```

Other command line options are identical to those of `arcsub`.

2.2.3 arccat

It is often useful to monitor the job progress by checking what it prints on the standard output or error. The command `arccat` assists here, extracting the corresponding information from the execution cluster and dumping it on the user’s screen. It works both for running tasks and for the finished ones. This allows a user to check the output of the finished task without actually retreiving it.

arccat [options] [job ...]

Options:

`-a, --all`		all jobs
`-j, --joblist`	*filename*	the file storing information about active jobs (default /.arc/jobs.dat)
`-i, --jobids-from-file`	*filename*	file containing a list of job IDs
`-s, --status`	*statusstr*	only select jobs whose status is *statusstr*
`-o, --stdout`		show the stdout of the job (default)
`-e, --stderr`		show the stderr of the job
`-l, --joblog`		show the CE’s error log of the job
`-P, --listplugins`		list the available plugins
`-t, --timeout`	*time*	timeout for queries (default 20 sec)
`-d, --debug`	*verbosity*	verbosity level is one of FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG

`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page
Options in ARC 11.05:		
`-c, --cluster`	`[-]`*url*	explicitly select or reject (-) a specific site
Options in ARC 12.05:		
`-c, --cluster`	*designator*	select one or more computing elements by an alias for a single CE, a group of CEs or a URL
`-r, --rejectmanagement`	*URL*	skip jobs which are on a computing element with a given URL
Arguments:		
`job ...`		list of job IDs and/or job names

The `arccat` command returns the standard output of a job (`-o` option), the standard error (`-e` option) or errors reported by either Grid Manager or A-REX (`-l` option).

Other command line options have the same meaning as in `arcstat`.

> When several of the `-c <cluster>`, `-i <jobidfile>` and `[job...]` command line options are specified, the command prints logs of all jobs listed on the command line plus all jobs on the specified clusters plus all jobs from the specified `jobidfile`.
>
> However the `-c -<cluster>` (for ARC 11.05), `-r <URL>` (for ARC 12.05) and `-s <status>` options will filter the jobs selected by the above mentioned options, or if none of those are specified, then these will filter all the jobs.
>
> For example, `arccat -s Finished -c mycluster <jobid>` will print logs of the finished jobs on `mycluster` plus of `<jobid>` but only if it is finished. Or, `arccat -i jobidfile -r mycluster.example.org` will print logs of jobs which are in the `jobidfile` but not on `mycluster.example.org`.

2.2.4 arcget

To retrieve the results of a finished job, the `arcget` command should be used. It will transfer the files specified for download in job description to the user's computer.

arcget [options] [job ...]

Options:		
`-a, --all`		all jobs
`-j, --joblist`	*filename*	the file storing information about active jobs (default /.arc/jobs.dat)
`-i, --jobids-from-file`	*filename*	file containing a list of job IDs
`-s, --status`	*statusstr*	only select jobs whose status is *statusstr*
`-D, --dir`	*dirname*	download path (the job directory will be created in that location)
`-J, --usejobname`	*dirname*	use the job name instead of the short ID as the job directory name

`-k, --keep`		keep files in the Grid (do not clean)
`-f, --force`		force download (overwrite existing job directory)
`-P, --listplugins`		list the available plugins
`-t, --timeout`	*time*	timeout for queries (default 20 sec)
`-d, --debug`	*verbosity*	verbosity level is one of FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG
`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page
Options in ARC 11.05:		
`-c, --cluster`	`[-]`*name*	explicitly select or reject a specific site (cluster)
Options in ARC 12.05:		
`-c, --cluster`	*designator*	select one or more computing elements by an alias for a single CE, a group of CEs or a URL
`-r, --rejectmanagement`	*URL*	skip jobs which are on a computing element with a given URL
Arguments:		
`job ...`		list of job IDs and/or job names

Only the results of jobs that have finished can be downloaded. Just like in `arcstat` and `arccat` cases, the job can be referred to either by the job ID that was returned by `arcsub` at submission time, or by its name, if the job description contained a job name attribute.

By default, the job is downloaded into a newly created directory in the current path, with the name typically being a large random string. In order to instruct `arcget` to use another path, use option `-D` (note the capital "D"), e.g.

```
arcget -D /tmp/myjobs "Test job nr 1"
```

> After downloading, your jobs will be erased from the execution site! Use command line option `-k` to keep finished jobs in the Grid.

Other command line options are identical to those of e.g. `arcstat`.

> When several of the `-c <cluster>`, `-i <jobidfile>` and `[job...]` command line options are specified, the command retrieves all jobs listed on the command line plus all jobs on the specified clusters plus all jobs from the specified `jobidfile`.
>
> However the `-c -<cluster>` (for ARC 11.05), `-r <URL>` (for ARC 12.05) and `-s <status>` options will filter the jobs selected by the above mentioned options, or if none of those are specified, then these will filter all the jobs.
>
> For example, `arcget -s Finished -c mycluster <jobid>` will retrieve the finished jobs on `mycluster` plus `<jobid>` but only if it is finished. Or, `arcget -i jobidfile -r mycluster.example.org` will retrieve jobs which are in the `jobidfile` but not on `mycluster.example.org`.

2.2.5 arcsync

It is advised to start every grid session by running `arcsync`, especially when changing workstations. The reason is that your job submission history is cached on your machine, and if you are using ARC client

installations on different machines, your local lists of submitted jobs will be different. To synchronise these lists with the information in the Information System, use the `arcsync` command.

arcsync [options]

Options:

`-j, --joblist`	*filename*	the file storing information about active jobs (default /.arc/jobs.dat)
`-f, --force`		don't ask for confirmation
`-T, --truncate`		truncate the job list before synchronising
`-P, --listplugins`		list the available plugins
`-t, --timeout`	*seconds*	timeout for network communication, in seconds (default 20)
`-d, --debug`	*verbosity*	verbosity level, FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG - default WARNING
`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page
Options in ARC 11.05:		
`-c, --cluster`	`[-]`*name*	explicitly select or reject a specific site
`-g, --index`	*url*	explicitly select or reject (-) a specific index server
Options in ARC 12.05:		
`-c, --cluster`	*designator*	select one or more computing elements by an alias for a single CE, a group of CEs or a URL
`-g, --index`	*designator*	select one or more registries by an alias for a single registry, a group of registries or a URL
`-R, --rejectdiscovery`	*URL*	skip the service with the given URL during service discovery

The ARC client keeps a local list of jobs in the user's home directory. If this file is lost, corrupt, or the user wants to recreate the file on a different workstation, the `arcsync` command will recreate this file from the information available in the Information System.

Since the information about a job retrieved from a cluster can be slightly out of date if the user very recently submitted or removed a job, a warning is issued when this command is run. The `-f` option disables this warning.

If the job list is not empty when invoking syncronisation, the old jobs will be merged with the new jobs, unless the `-T` option is given (note the capital "T"), in which case the job list will first be truncated and then the new jobs will be added.

2.2.6 arcinfo

The `arcinfo` command is used to obtain status information about clusters on the Grid.

arcinfo [options]

Options:

`-l, --long`		long format (extended information)
`-L, --list-configured-services`		print a list of services configured in the client.conf
`-P, --listplugins`		list the available plugins
`-t, --timeout`	*seconds*	timeout for network communication, in seconds (default 20)
`-d, --debug`	*verbosity*	verbosity level, FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG - default WARNING
`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page
Options in ARC 11.05:		
`-c, --cluster`	`[-]`*name*	explicitly select or reject a specific site
`-g, --index`	*url*	explicitly select or reject (-) a specific index server
Options in ARC 12.05:		
`-c, --cluster`	*designator*	select one or more computing elements by an alias for a single CE, a group of CEs or a URL
`-g, --index`	*designator*	select one or more registries by an alias for a single registry, a group of registries or a URL
`-R, --rejectdiscovery`	*URL*	skip the service with the given URL during service discovery
`-S, --submissioninterface`	*InterfaceName*	only get information about executon targets which supports this job submission interface (e.g. org.nordugrid.gridftpjob, org.ogf.glue.emies.activitycreation, org.ogf.bes)
Options in ARC 13.02:		
`-I, --infointerface`	*InterfaceName*	the computing element specified by URL at the command line should be queried using this information interface (possible options: org.nordugrid.ldapng, org.nordugrid.ldapglue2, org.nordugrid.wsrfglue2, org.ogf.glue.emies.resourceinfo)

The `arcinfo` command is used to obtain information about clusters and queues (*targets*) available on the Grid. Either the `--cluster` or `--index` flag should be used to specify the target(s) which should be queried for information. Both of these flags take a service endpoint as argument. See `arcsub` and the configuration notes in Section 4 for description of these.

Detailed information about queried computing services can be obtained by specifying the `--long` flag.

When specifying the `--index` flag, the information about the computing services registered at the index server will be queried, rather than the status of the index server itself.

2.2.7 arckill

It happens that a user may wish to cancel a job. This is done by using the `arckill` command. A job can be killed almost at any stage of processing through the Grid.

arckill [options] [job ...]

Options:

`-a, --all`		all jobs
`-j, --joblist`	*filename*	the file storing information about active jobs (default /.arc/jobs.dat)
`-i, --jobids-from-file`	*filename*	file containing a list of job IDs
`-s, --status`	*statusstr*	only select jobs whose status is *statusstr*
`-k, --keep`		keep files in the Grid (do not clean)
`-P, --listplugins`		list the available plugins
`-t, --timeout`	*time*	timeout for queries (default 20 sec)
`-d, --debug`	*verbosity*	verbosity level is one of FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG
`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page
Options in ARC 11.05:		
`-c, --cluster`	`[-]`*url*	explicitly select or reject (-) a specific site
Options in ARC 12.05:		
`-c, --cluster`	*designator*	select one or more computing elements by an alias for a single CE, a group of CEs or a URL
`-r, --rejectmanagement`	*URL*	skip jobs which are on a computing element with a given URL

Arguments:

`job ...`	list of job IDs and/or job names

If a job is killed, its traces are being cleaned from the Grid. If you wish to keep the killed job in the system, e.g. for a post-mortem analysis, use the `-k` option.

> Job cancellation is an asynchronous process, such that it may take a few minutes before the job is actually cancelled.

Command line options have the same meaning as the corresponding ones of `arcstat` and others.

> When several of the `-c <cluster>`, `-i <jobidfile>` and `[job...]` command line options are specified, the command kills all jobs listed on the command line plus all jobs on the specified clusters plus all jobs from the specified `jobidfile`.
>
> However the `-c -<cluster>` (for ARC 11.05), `-r <URL>` (for ARC 12.05) and `-s <status>` options will filter the jobs selected by the above mentioned options, or if none of those are specified, then these will filter all the jobs.
>
> For example, `arckill -s INLRMS:R -c mycluster <jobid>` will kill the running jobs on `mycluster` plus `<jobid>` but only if it is running. Or, `arckill -i jobidfile -r mycluster.example.com` will kill all jobs which are in the `jobidfile` but not on `mycluster.example.com`.

2.2.8 arcclean

If a job fails or gets killed with `-k` option, or when you are not willing to retrieve the results for some reasons, a good practice for users is not to wait for the system to clean up the job leftovers, but to use

`arcclean` to release the disk space and to remove the job ID from the list of submitted jobs and from the Information System.

arcclean [options] [job ...]

Options:

`-a, --all`		all jobs
`-j, --joblist`	*filename*	the file storing information about active jobs (default /.arc/jobs.dat)
`-i, --jobids-from-file`	*filename*	file containing a list of job IDs
`-s, --status`	*statusstr*	only select jobs whose status is *statusstr*
`-f, --force`		removes the job ID from the local list even if the job is not found on the Grid
`-P, --listplugins`		list the available plugins
`-t, --timeout`	*time*	timeout for queries (default 20 sec)
`-d, --debug`	*verbosity*	verbosity level is one of FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG
`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page
Options in ARC 11.05:		
`-c, --cluster`	`[-]`*name*	explicitly select or reject a specific site (cluster)
Options in ARC 12.05:		
`-c, --cluster`	*designator*	select one or more computing elements by an alias for a single CE, a group of CEs or a URL
`-r, --rejectmanagement`	*URL*	skip jobs which are on a computing element with a given URL

Arguments:

`job ...`	list of job IDs and/or job names

Only jobs that have finished or were cancelled can be cleaned.

It happens ever so often that the job is cleaned by the system, or is otherwise unreachable, and yet your local job list file still has it listed. Use `-s` option with value `Undefined` to remove such stale job information from the local list. Note that specifying `-a` and `-f` options together also removes such stale job information, while also removing finished and cancelled jobs.

Other command line options have the same meaning as the corresponding ones of `arcstat` and others.

When several of the `-c <cluster>`, `-i <jobidfile>` and `[job...]` command line options are specified, the command cleans all jobs listed on the command line plus all jobs on the specified clusters plus all jobs from the specified `jobidfile`.

However the `-c -<cluster>` (for ARC 11.05), `-r <URL>` (for ARC 12.05) and `-s <status>` options will filter the jobs selected by the above mentioned options, or if none of those are specified, then these will filter all the jobs.

For example, `arcclean -s FAILED -c mycluster <jobid>` will clean the failed jobs on `mycluster` plus `<jobid>` but only if it is failed. Or, `arcclean -i jobidfile -r mycluster.example.com` will

clean all jobs which are in the `jobidfile` but not on `mycluster.example.com`.

2.2.9 arcrenew

Quite often, the user proxy expires while the job is still running (or waiting in a queue). In case such job has to upload output files to a Grid location (a storage element), it will fail. By using the `arcrenew` command, users can upload a new proxy to the job. This can be done while a job is still running, thus preventing it from failing

If a job has failed in file upload due to expired proxy, `arcrenew` can be issued whithin 24 hours (or whatever is the expiration time set by the site) after the job end, which must be followed by `arcresume`. The Grid Manager or A-REX will then attempt to finalize the job by uploading the output files to the desired location.

arcrenew [options] [job ...]

Options:

`-a, --all`		all jobs
`-j, --joblist`	*filename*	the file storing information about active jobs (default /.arc/jobs.dat)
`-i, --jobids-from-file`	*filename*	file containing a list of job IDs
`-s, --status`	*statusstr*	only select jobs whose status is *statusstr*
`-P, --listplugins`		list the available plugins
`-t, --timeout`	*time*	timeout for queries (default 20 sec)
`-d, --debug`	*verbosity*	verbosity level is one of FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG
`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page
Options in ARC 11.05:		
`-c, --cluster`	`[-]`*name*	explicitly select or reject a specific site (cluster)
Options in ARC 12.05:		
`-c, --cluster`	*designator*	select one or more computing elements by an alias for a single CE, a group of CEs or a URL
`-r, --rejectmanagement`	*URL*	skip jobs which are on a computing element with a given URL

Arguments:

`job ...`	list of job IDs and/or job names

Prior to using `arcrenew`, be sure to actually create the new proxy by running `arcproxy`!

Command line options have the same meaning as the corresponding ones of `arcstat` and others.

When several of the `-c <cluster>`, `-i <jobidfile>` and `[job...]` command line options are specified, the command renews proxies of all jobs listed on the command line plus all jobs on the specified clusters plus all jobs from the specified `jobidfile`.

However the `-c -<cluster>` (for ARC 11.05), `-r <URL>` (for ARC 12.05) and `-s <status>` options will filter the jobs selected by the above mentioned options, or if none of those are specified, then these will filter all the jobs.

For example, `arcrenew -s FAILED -c mycluster <jobid>` will renew proxies of the failed jobs on `mycluster` plus `<jobid>` but only if it is failed. Or, `arcrenew -i jobidfile -r mycluster.example.com` will renew proxies of all jobs which are in the `jobidfile` but not on `mycluster.example.com`.

2.2.10 arcresume

In some cases a user may want to restart a failed job, for example, when input files become available, or the storage element for the output files came back online, or when a proxy is renewed with `arcrenew`. This can be done using the `arcresume` command.

Make sure your proxy is still valid, or when uncertain, run `arcproxy` followed by `arcrenew` before `arcresume`. The job will be resumed from the state where it has failed.

arcresume [options] [job ...]

`-a, --all`		all jobs
`-j, --joblist`	*filename*	the file storing information about active jobs (default /.arc/jobs.dat)
`-i, --jobids-from-file`	*filename*	file containing a list of job IDs
`-s, --status`	*statusstr*	only select jobs whose status is *statusstr*
`-P, --listplugins`		list the available plugins
`-t, --timeout`	*time*	timeout for queries (default 20 sec)
`-d, --debug`	*verbosity*	verbosity level is one of FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG
`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page
Options in ARC 11.05:		
`-c, --cluster`	`[-]`*name*	explicitly select or reject a specific site (cluster)
Options in ARC 12.05:		
`-c, --cluster`	*designator*	select one or more computing elements by an alias for a single CE, a group of CEs or a URL
`-r, --rejectmanagement`	*URL*	skip jobs which are on a computing element with a given URL
Arguments:		
`job ...`		list of job IDs and/or job names

Command line options have the same meaning as the corresponding ones of `arcstat` and others.

When several of the `-c <cluster>`, `-i <jobidfile>` and `[job...]` command line options are specified, the command resumes all jobs listed on the command line plus all jobs on the specified clusters plus all jobs from the specified `jobidfile`.

However the `-c -<cluster>` (for ARC 11.05), `-r <URL>` (for ARC 12.05) and `-s <status>` options will filter the jobs selected by the above mentioned options, or if none of those are specified, then these will filter all the jobs.

For example, `arcresume -s FAILED -c mycluster <jobid>` will resume the failed jobs on `mycluster` plus `<jobid>` but only if it is failed. Or, `arcresume -i jobidfile -r mycluster.example.com` will resume all jobs which are in the `jobidfile` but not on `mycluster.example.com`.

2.2.11 arcresub

Quite often it happens that a user would like to re-submit a job, but has difficulties recovering the original job description file (e.g. the xRSL file). This happens when job description files are created by scripts on-fly, and matching of job description to the job ID is not straightforward. The utility called `arcresub` helps in such situations, allowing users to resubmit jobs.

arcresub [options] [job ...]

Options:

`-a, --all`		all jobs
`-j, --joblist`	*filename*	the file storing information about active jobs (default /.arc/jobs.dat)
`-i, --jobids-from-file`	*filename*	file containing a list of job IDs
`-o, --jobids-to-file`	*filename*	the IDs of the submitted jobs will be appended to this file
`-m, --same`		re-submit to the same site
`-M, --not-same`		do not resubmit to the same cluster
`-s, --status`	*statusstr*	only select jobs whose status is *statusstr*
`-k, --keep`		keep files in the Grid (do not clean)
`-b, --broker`	*string*	select broker method (default is Random)
`-P, --listplugins`		list the available plugins
`-t, --timeout`	*time*	timeout for queries (default 20 sec)
`-d, --debug`	*verbosity*	verbosity level is one of FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG
`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page
Options in ARC 11.05:		
`-g, --index`	*url*	explicitly select or reject (-) a specific index server
`-c, --cluster`	`[-]`*name*	explicitly select or reject a specific source site
`-q, --qluster`	`[-]`*name*	explicitly select or reject a specific site as resubmission target

Options in ARC 12.05:		
`-g, --index`	*designator*	select one or more registries by an alias for a single registry, a group of registries or a URL
`-c, --cluster`	*designator*	select one or more computing elements by an alias for a single CE, a group of CEs or a URL
`-q, --qluster`	*designator*	select one or more computing elements for the new jobs by an alias for a single CE, a group of CEs or a URL
`-r, --rejectmanagement`	*URL*	skip jobs which are on a computing element with a given URL
`-R, --rejectdiscovery`	*URL*	skip the service with the given URL during service discovery
`-S, --submissioninterface`	*InterfaceName*	only use this interface for submitting (e.g. org.nordugrid.gridftpjob, org.ogf.glue.emies.activitycreation, org.ogf.bes)
Options in ARC 13.02:		
`-I, --infointerface`	*InterfaceName*	the computing element specified by URL at the command line should be queried using this information interface (possible options: org.nordugrid.ldapng, org.nordugrid.ldapglue2, org.nordugrid.wsrfglue2, org.ogf.glue.emies.resourceinfo)
Arguments:		
`job ...`		list of job IDs and/or job names

More than one job ID and/or job name can be given. If several jobs were submitted with the same job name all those jobs will be resubmitted.

If the job description of a job to be resubmitted, contained any local input files, checksums of these was calculated and stored in the job list, and those will be used to check whether the files has changed. If local input files has changed the job will not be resubmitted.

In case the job description is not found in the job list, an attempt will be made to retrieve it from the cluster holding the orignal job. This however may fail, since both the submission client and the cluster can have made modifications to the job description.

Upon resubmision the job will receive a new job ID, and the old job ID will be stored in the local job list file, enabling future back tracing of the resubmitted job.

Upon resubmision the job will receive a new job ID. The old job ID will be kept in the local job list file, enabling future back tracing of the resubmitted job.

Regarding command line options, `arcresub` behaves much like `arcsub`, except that `-c` in this case indicates not the submission target site, but on the contrary, the **site from which the jobs will be resubmitted**. Submission target site is specified with option `-q`. If you wish to re-submit each job to the same site, use option `-m`.

If the original job was successfully killed, its traces will be removed from the execution site, unless the `-k` option is specified.

When several of the `-c <cluster>`, `-i <jobidfile>` and `[job...]` command line options are specified, the command resubmits all jobs listed on the command line plus all jobs on the specified clusters plus all jobs from the specified `jobidfile`.

However the `-c -<cluster>` (for ARC 11.05), `-r <URL>` (for ARC 12.05) and `-s <status>` options will filter the jobs selected by the above mentioned options, or if none of those are specified, then these will filter all the jobs.

For example, `arcresub -s FAILED -c mycluster <jobid>` will resubmit the

running jobs on `mycluster` plus `<jobid>` but only if it is failed. Or, `arcresub -i jobidfile -r mycluster.example.org` will resubmit all jobs which are in the `jobidfile` but not on `mycluster.example.org`.

2.3 Data manipulation

ARC provides basic data management tools to copy, create, list, rename and remove files and directories to, from and between Grid storage elements and index services.

2.3.1 arcls

`arcls` is a simple utility that allows to list contents and view some attributes of objects of a specified (by a URL) remote directory.

arcls [options] <URL>

Options:

`-l, --long`		detailed listing
`-L, --locations`		detailed listing including URLs from which files can be downloaded
`-m, --metadata`		display all available metadata
`-r, --recursive`		operate recursively (if possible)
`-D, --depth`	*recursion_level*	operate recursively (if possible) up to specified level (0 - no recursion) (available in ARC 13.02)
`-n, --nolist`		show only description of requested object, do not list content of directories (like ls -d).
`-f, --forcelist`		treat requested object as directory and always try to list content.
`-c, --checkaccess`		check readability of object. Retrieving and showing information about object is supressed.
`-t, --timeout`	*seconds*	timeout for network communication, in seconds (default 20)
`-P, --plugins`		list the available plugins (protocols supported)
`-d, --debug`	*verbosity*	verbosity level, FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG - default WARNING
`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page

Arguments:

`URL`	file or directory URL

This tool is very convenient not only because it allows to list files at a Storage Element or records in an indexing service, but also because it can give a quick overview of a job's working directory, which is explicitly given by job ID.

Usage examples can be as follows:

```
arcls -l gsiftp://lscf.nbi.dk:2811/jobs/1323842831451666535
arcls srm://grid.uio.no:8446/srm/managerv2?SFN=/johndoe/log2
```

Examples of URLs accepted by this tool can be found in Section 3.

2.3.2 arccp

`arccp` is a powerful tool to copy files over the Grid. It is a part of the A-REX, but can be used by the User Interface as well.

arccp [options] <source> <destination>

Options:

`-p, --passive`		use passive transfer (off by default if secure is on, on by default if secure is not requested)
`-n, --nopassive`		do not try to force passive transfer
`-f, --force`		if the destination is an indexing service and not the same as the source and the destination is already registered, then the copy is normally not done. However, if this option is specified the source is assumed to be a replica of the destination created in an uncontrolled way and the copy is done like in case of replication. Using this option also skips validation of completed transfers.
`-i, --indicate`		show progress indicator. If the transfer time is short then there may be no indicator.
`-T, --notransfer`		do not transfer file, just register it - destination must be non-existing meta-url
`-u, --secure`		use secure transfer (insecure by default)
`-y, --cache`	*path*	path to local cache (use to put file into cache). See [4] for information on caching.
`-r, --recursive`		operate recursively (if possible)
`-D, --depth`	*recursion_level*	operate recursively (if possible) up to specified level (0 - no recursion) (available in ARC 13.02)
`-R, --retries`	*number*	how many times to retry transfer of every file before failing
`-L, --location`	*URL*	physical file to write to when destination is an indexing service. Must be specified for indexing services which do not automatically generate physical locations. Can be specified multiple times - locations will be tried in order until one succeeds.
`-3, --thirdparty`		perform third party transfer, where the destination pulls from the source (only available with GFAL plugin and ARC 13.02)
`-t, --timeout`	*seconds*	timeout for network communication, in seconds (default 20)
`-P, --plugins`		list the available plugins (protocols supported)
`-d, --debug`	*verbosity*	verbosity level, FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG - default WARNING

`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page
Arguments:		
`source`		source URL
`destination`		destination URL

This command transfers contents of a file between 2 end-points. End-points are represented by URLs or meta-URLs or local file paths. For supported endpoints please refer to Section 3.

`arccp` can perform multi-stream transfers if `threads` URL option is specified and server supports it.

Source can end with `"/"`. In that case, the set of files under source will be copied into destination and destination must also end with `"/"`. Destination will be created if it does not exist. If copying deeper than one level is required then `-r` or `-D` must be used. If destination alone ends with `"/"`, it is extended with the part of source after last `"/"`, thus allowing users to skip the destination file or directory name if it is meant to be identical to the source.

Usage examples of `arccp` are:

```
arccp -i gsiftp://lscf.nbi.dk:2811/jobs/1323842831451666535/job.out job.out
arccp http://www.nordugrid.org/data/somefile gsiftp://hathi.hep.lu.se/data/
arccp gsiftp://pgs02.grid.upjs.sk:2811/jobs/13331297786445657047863/ output/
arccp -R 3 my.file srm://srm.host.org;spacetoken=MYTOKEN/my.file.1
```

2.3.3 arcrm

The `arcrm` command allows users to erase files and directories at any location specified by a valid URL.

arcrm [options] <URL> [URL ...]

Options:		
`-f, --force`		remove logical file name registration even if not all physical instances were removed
`-t, --timeout`	*seconds*	timeout for network communication, in seconds (default 20)
`-P, --plugins`		list the available plugins (protocols supported)
`-d, --debug`	*verbosity*	verbosity level, FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG - default WARNING
`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page
Arguments:		
`URL [URL ...]`		file or directory URL (mutliple URLs are supported in ARC 13.02 and above)

A convenient use for `arcrm` is to erase the files in a data indexing catalog, as it will not only remove the physical instance, but also will clean up the database record.

Here is an `arcrm` example:

```
arcrm srm://grid.uio.no/grid/atlas/AOD_0947.pool.root
```

2.3.4 arcmkdir

The `arcmkdir`[‡] command allows users to create directories, if the protocol of the specified URL supports it.

arcmkdir [options] <URL>

Options:

`-p, --parents`		make parent directories as needed
`-t, --timeout`	*seconds*	timeout for network communication, in seconds (default 20)
`-P, --plugins`		list the available plugins (protocols supported)
`-d, --debug`	*verbosity*	verbosity level, FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG - default WARNING
`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page

Arguments:

`URL`	directory to create

`arcmkdir` creates directories on grid storage elements and indexing services. If the parent directory does not exist and -p is not specified, then arcmkdir will probably fail, but it depends on the protocol. The permissions on the new directory are the default of the server, or if the protocol requires them to be specified then the directory is only readable/writable/searchable by the user (the equivalent of 700 on a file system).

Example:

```
arcmkdir srm://grid.uio.no/grid/atlas/newdir
```

2.3.5 arcrename

The `arcrename`[§] command allows users to rename files and directories, if the protocol of the specified URL supports it.

arcrename [options] <old URL> <new URL>

Options:

`-t, --timeout`	*seconds*	timeout for network communication, in seconds (default 20)
`-P, --plugins`		list the available plugins (protocols supported)
`-d, --debug`	*verbosity*	verbosity level, FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG - default WARNING

[‡]only available in version 12.05 and later
[§]only available in version 13.02 and later

`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page
Arguments:		
`old URL`		current name of file or directory
`new URL`		new name for file or directory

The `arcrename` command renames files or directories on grid storage elements and indexing services. The path component of old URL and new URL must differ and it must be the only component of both URLs which is different. arcrename will exit with an error if the paths are equivalent or other components of the URLs are different. Renaming a URL to an existing URL will either fail or overwrite the existing URL, depending on the protocol.

2.4 Test suite

2.4.1 arctest

`arctest` is a simple utility that tests very basic functionalities of the middleware. It is convenient for:

- first-time users who do not know job description languages and yet want to test e.g. their credentials or client setup,
- system administrators who'd like to quickly test their installations without having to learn job description languages.

The `arctest` utility contains pre-defined test jobs which can be submitted either to a specific test site or to a regular Grid infrastructure. In addition, `arctest` can provide basic information about available user credentials (proxy certificate).

arctest [options]

Options:

`-J, --jobid`	*integer*	test job number
`-j, --joblist`	*filename*	the file storing information about active jobs (on Linux, default $/.arc/jobs.dat)
`-o, --jobids-to-file`	*filename*	the IDs of the submitted jobs will be appended to this file
`-D, --dryrun`		add dryrun option to the job description
`-x, --dumpdescription`		do not submit – dump transformed job description to stdout
`-E, --certificate`		prints information about available user credentials
`-b, --broker`	*string*	select broker method (default is Random)
`-t, --timeout`	*seconds*	timeout for network communication, in seconds (default 20)
`-d, --debug`	*verbosity*	verbosity level, FATAL, ERROR, WARNING, INFO, VERBOSE or DEBUG - default WARNING
`-z, --conffile`	*filename*	configuration file (on Linux, default $HOME/.arc/client.conf)

`-v, --version`		print version information
`-h, --help`		print help page
Options in ARC 11.05:		
`-c, --cluster`	`[-]`*url*	explicitly select or reject (-) a specific site
`-g, --index`	`[-]`*url*	explicitly select or reject (-) a specific index server
Options in ARC 12.05:		
`-c, --cluster`	*designator*	select one or more computing elements by an alias for a single CE, a group of CEs or a URL
`-g, --index`	*designator*	select one or more registries by an alias for a single registry, a group of registries or a URL
`-R, --rejectdiscovery`	*URL*	skip the service with the given URL during service discovery
`-S, --submissioninterface`	*InterfaceName*	only use this interface for submitting (e.g. org.nordugrid.gridftpjob, org.ogf.glue.emies.activitycreation, org.nordugrid.xbes)
Options in ARC 13.02:		
`-I, --infointerface`	*InterfaceName*	the computing element specified by URL at the command line should be queried using this information interface (possible options: org.nordugrid.ldapng, org.nordugrid.ldapglue2, org.nordugrid.wsrfglue2, org.ogf.glue.emies.resourceinfo)
`-r, --runtime`	*int*	test job 1 runtime in minutes

There are currently three test jobs defined. Once submitted, their results can be inspected and retrieved in a usual manner, using `arccat`, `arcstat`, `arcget` etc..

Test job descriptions:

1. **ARC 11.05 and 12.05:** A classical "Hello World" job, printing *hello, grid* text to standard output at a remote execution site.
   ARC 13.02: The job calculates prime-numbers for a number of minutes given by -r (default 5) and outputs the list to stderr. The source-code for the prime-number program, the Makefile and the executable are downloaded to the cluster from HTTP and FTP servers and the program is compiled before running.
2. Lists all environment variables defined for the Grid user at the remote site (using standard output).
3. Downloads the pre-defined input file from the HTTP server, and produces an output file by copying input with a new name. This job thus demonstrates usage of `inputfiles` and `outputfiles` attributes of job description languages.

`arctest` is complementary to the `arcinfo` utility, which extracts information about Grid resources without submitting test jobs, and to `arcproxy -I`, which provides more detailed information about user credentials.

Chapter 3

URLs

File locations in ARC can be specified both as local file names, and as Internet standard *Uniform Resource Locators (URL)*. There are also some additional URL *options* that can be used.

Depending on the installed ARC components some or all of the following transfer protocols and metadata services are supported:

`ftp`	ordinary *File Transfer Protocol (FTP)*
`gsiftp`	GridFTP, the Globus® -enhanced FTP protocol with security, encryption, etc. developed by The Globus Alliance [3]
`http`	ordinary *Hyper-Text Transfer Protocol (HTTP)* with PUT and GET methods using multiple streams
`https`	HTTP with SSL v3
`httpg`	HTTP with Globus® GSI
`ldap`	ordinary *Lightweight Data Access Protocol (LDAP)* [8]
`srm`	Storage Resource Manager (SRM) service [6]
`root`	Xrootd protocol (available in ARC 2.0.0 and later (read-only), 4.2.0 and later (full functionality))
`rucio`	Next generation ATLAS data management system (read only, available in ARC 4.1.0 and later)
`acix`	ARC Cache Index (read only, available in ARC 4.1.0 and later)
`file`	local to the host file name with a full path

An URL can be used in a standard form, i.e.

```
protocol://[host[:port]]/file
```

Or, to enhance the performance or take advantage of various features, it can have additional options:

```
protocol://[host[:port]][;option[;option[...]]]/file[:metadataoption[:metadataoption[...]]
```

For a metadata service URL, construction is the following:

```
protocol://[url[|url[...]]@]host[:port][;option[;option[...]]]
     /lfn[:metadataoption[:metadataoption[...]]]
```

where the nested URL(s) are physical replicas. Options are passed on to all replicas, but if it is desired to use the same option with a different value for all replicas, the option can be specified as a common option using the following syntax:

```
protocol://[;commonoption[;commonoption]|][url[|url[...]]@]host[:port]
     [;option[;option[...]]/lfn[:metadataoption[:metadataoption[...]]]
```

In user-level tools, URLs may be expressed in this syntax, or there may simpler ways to contruct complex URLs. In particular, command line tools such as `arccp` and the xRSL and JSDL job description languages provide methods to express URLs and options in a simpler way.

For the SRM service, the syntax is

```
srm://host[:port][;options]/[service_path?SFN=]file[:metadataoptions]
```

Versions 1.1 and 2.2 of the SRM protocol are supported. The default *service_path* is srm/managerv2 when the server supports v2.2, srm/managerv1 otherwise.

For Rucio the URLs look like

```
rucio://rucio-lb-prod.cern.ch/replicas/scope/lfn
```

The Rucio authorisation URL can be specified with the environment variable `$RUCIO_AUTH_URL`. The Rucio account to use can be specified either through the rucioaccount URL option or `$RUCIO_ACCOUNT` environment variable. If neither are specified the account is taken from the VOMS nickname attribute.

For ACIX the URLs look like

```
acix://cacheindex.ndgf.org:6443/data/index?url=http://host.org/file1
```

The URL components are:

`host[:port]`	Hostname or IP address [and port] of a server
`lfn`	Logical File Name
`url`	URL of the file as registered in indexing service
`service_path`	End-point path of the web service
`file`	File name with full path
`option`	URL option
`commonoption`	URL option for all replicas
`metadataoption`	Metadata option

The following URL options are supported:

`threads=<number>`	specifies number of parallel streams to be used by GridFTP or HTTP(s,g); default value is 1, maximal value is 10
`exec=yes|no`	means the file should be treated as executable
`preserve=yes|no`	specify if file must be uploaded to this destination even if job processing failed (default is `no`)
`cache=yes|no|renew|copy|check|invariant`	indicates whether the file should be cached; default for input files in A-REX is `yes`. `renew` forces a download of the file, even if the cached copy is still valid. `copy` forces the cached file to be copied (rather than linked) to the session dir, this is useful if for example the file is to be modified. `check` forces a check of the permission and modification time against the original source. `invariant` disables checking the original source modification time. (`check` option is available in ARC 2.0.0 and above, `invariant` option is available in ARC 3.0.0 and above).

`readonly=yes|no`	for transfers to `file://` destinations, specifies whether the file should be read-only (unmodifiable) or not; default is `yes`
`secure=yes|no`	indicates whether the GridFTP data channel should be encrypted; default is `no`
`blocksize=<number>`	specifies size of chunks/blocks/buffers used in GridFTP or HTTP(s,g) transactions; default is protocol dependent
`checksum=cksum|md5|` `adler32|no`	specifies the algorithm for checksum to be computed (for transfer verification or provided to the indexing server). This is overridden by any metadata options specified (see below). If this option is not provided, the default for the protocol is used. `checksum=no` disables checksum calculation.
`overwrite=yes|no`	make software try to overwrite existing file(s), i.e. before writing to destination, tools will try to remove any information/content associated with specified URL
`protocol=gsi|gssapi|ssl|` `tls|ssl3`	to distinguish between different kinds of `https`/`httpg` and `srm` protocols. Here `gssapi` stands for `httpg` implemention using only GSSAPI functions to wrap data and `gsi` uses additional headers as implmented in Globus IO. The `ssl` and `tls` stand for usual `https` and especially usable only if used with `srm` protocol. The `ssl3` is mostly same as `ssl` but uses SSLv3 hadshake while establishing `https` connection. The default is `gssapi` for `srm` connections, `tls` for `https` and `gssapi` for `httpg`. In case of `srm` if default fails, `gsi` is then tried.
`spacetoken=<pattern>`	specify the space token to be used for uploads to SRM storage elements supporting SRM version 2.2 or higher
`autodir=yes|no`	specify if before writing to specified location software should try to create all directories mentioned in specified URL. Currently this applies to FTP and GridFTP only. Default for those protocols is `yes`
`tcpnodelay=yes|no`	controls the use of the TCP_NODELAY socket option (which disables the Nagle algorithm). Applies to http(s) only. Default is `no` (supported only in `arcls` and other `arc*` tools)
`transferprotocol=protocols`	specify transfer protocols for meta-URLs such as SRM. Multiple protocols can be specified as a comma-separated list in order of preference.
`rucioaccount=account`	specify the Rucio account to use when authenticating with Rucio.
`httpputpartial=yes|no`	while storing file on HTTP(S) server software will try to send it in chunks/parts. If server reports error for partial PUT command software will fall back to tranfering file in single piece. This behavior is non-standard and not all servers report error properly. Hence default is safer 'no'.
`httpgetpartial=yes|no`	while retrieving file from HTTP(S) server software will try to read in in chunks/parts. If server does not support partial GET command it usually ignores request for partial transfer range and file is transfered in one piece. Default is 'yes'.

Local files are referred to by specifying either a location relative to the job submission working directory, or by an absolute path (the one that starts with "/"), preceded with a `file://` prefix.

URLs also support metadata options which can be used for registering additional metadata attributes or querying the service using metadata attributes. These options are specified at the end of the LFN and consist of name and value pairs separated by colons. The following attributes are supported:

`checksumtype`	Type of checksum. Supported values are cksum (default), md5 and adler32
`checksumvalue`	The checksum of the file

The checksum attributes may also be used to validate files that were uploaded to remote storage.

Examples of URLs are:

```
http://grid.domain.org/dir/script.sh
gsiftp://grid.domain.org:2811;threads=10;secure=yes/dir/input_12378.dat
ldap://grid.domain.org:389/lc=collection1,rc=Nordugrid,dc=nordugrid,dc=org
file:///home/auser/griddir/steer.cra
srm://srm.domain.org/griddir/user/file1:checksumtype=adler32:checksumvalue=12345678[1]
srm://srm.domain.org;transferprotocol=https/data/file2[2]
```

[1]This is a destination URL. The file will be copied to `srm.domain.org` at the path `griddir/user/file1` and the checksum will be compared to what is reported by the SRM service after the transfer.

[2]This is a source or destination URL. When getting a TURL from SRM the HTTPS transfer protocol will be requested.

Chapter 4

ARC Client Configuration

The default behaviour of an ARC client can be configured by specifying alternative values for some parameters in the client configuration file. The file is called `client.conf` and is located in directory `.arc` in user's home area, e.g., on Linux:

$HOME/.arc/client.conf

If this file is not present or does not contain the relevant configuration information, the global configuration files (if exist) or default values are used instead. Some client tools may be able to create the default `$HOME/.arc/client.conf` on Linux, if it does not exist.

The ARC configuration file consists of several configuration blocks. Each configuration block is identified by a keyword and contains configuration options for a specific part of the ARC middleware.

The configuration file is written in a plain text format known as INI. Configuration blocks start with identifying keywords inside square brackets. Typically, first comes a common block: `[common]`. Thereafter follows one or more attribute-value pairs written one on each line in the following format:

```
[common]
attribute1=value1
attribute2=value2
attribute3=value3 value4
# comment line 1
# comment line 2
...
```

Note that values must not be enclosed in quotes! Most attributes have counterpart command line options. Command line options always overwrite configuration attributes.

Client configuration is **different** for ARC 11.05 (client versions 1.*.*) and ARC 12.05 (client versions 2.*.*) and above.

Newer clients will work with the old configuration. Older clients will not understand most new configuration options.

In addition, configuration files from ARC 0.* versions will **not** work with newer client versions. The *ngclient2arc* tool is provided to help with migrating configuration to the new format.

ARC 11.05 clients recognise two configuration blocks, `[common]` and `[alias]`.

ARC 12.05 and above still makes use of the `[common]` block, but does not recognise `defaultservices` and `rejectservices` options, instead using `rejectdiscovery`, `rejectmanagement`, `infointerface` and

`submissioninterface` options. In addition, each service in ARC 12.05 and above has its own block: `[registry/<alias>]` for registry services and `[computing/<alias>]` for computing services, where the `<alias>` has to be a unique name for the service (without spaces).

In ARC versions 0.*, all configured services were used by default, but from version 11.05 default services must be explicitly specified in the configuration, either by the `defaultservices` (11.05) or `default` attributes (12.05 and above).

4.1 Block [common]

defaultservices (only in ARC 11.05)

This attribute is multi-valued.

This attribute is used to specify default services to be used. Defining such in the user configuration file will override the default services set in the system configuration.

The value of this attribute should follow the format:

```
service_type:grid:service_url
```

where `service_type` is type of service (e.g. `computing` or `index`), `grid` specifies type of middleware plugin to use when contacting the service (e.g. ARC0, ARC1, CREAM, etc.) and `service_url` is the URL used to contact the service. Several services can be listed, separated with a blank space (no line breaks allowed).

Example:

```
defaultservices=index:ARC0:ldap://index1.ng.org:2135/Mds-Vo-name=testvo,o=grid
␣index:ARC1:https://index2.ng.org:50000/isis
␣computing:ARC1:https://ce.arc.org:60000/arex
␣computing:CREAM:ldap://ce.glite.org:2170/o=grid
```

rejectservices (only in ARC 11.05)

This attribute is multi-valued.

This attribute can be used to indicate that a certain service should be rejected ("blacklisted"). Several services can be listed, separated with a blank space (no line breaks allowed).

Example: `rejectservices=computing:ARC1:https://bad.service.org/arex`

rejectdiscovery (since ARC 12.05)

Domain name of a service which should be rejected during service discovery. Jobs will not be submitted to any interface of that service. Multiple **rejectdiscovery** commands can be used.

Example:

```
rejectdiscovery=bad.server.org
rejectdiscovery=bad2.server.org
```

rejectmanagement (since ARC 12.05)

During job management operations, the jobs belonging to this service will be skipped. Multiple **rejectmanagement** commands can be used.

Example:

```
rejectmanagement=bad.server.org
rejectmanagement=bad2.server.org
```

infointerface (since ARC 12.05)

Default information interface used by service discovery and status query operations. If a service has no `infointerface` specified in its configuration block (see Section 4.2), this will be used by default.

Example:

```
infointerface=org.nordugrid.ldapng
```

submissioninterface (since ARC 12.05)

Default submission interface used by job management operations. If a service has no `submissioninterface` specified in its configuration block (see Section 4.2), this will be used by default.

Example:

```
submissioninterface=org.nordugrid.gridftpjob
```

verbosity

Default verbosity (debug) level to use for the ARC clients. Corresponds to the `-d` command line option of the clients. Default value is `WARNING`, possible values are `FATAL`, `ERROR`, `WARNING`, `INFO`, `VERBOSE` or `DEBUG`.

Example: `verbosity=INFO`

timeout

Sets the period of time the client should wait for a service (information, computing, storage etc) to respond when communicating with it. The period should be given in seconds. Default value is 20 seconds. This attribute corresponds to the `-t` command line option.

Example: `timeout=10`

brokername

Configures which brokering algorithm to use during job submission. This attribute corresponds to the `-b` command line option. The default one is the `Random` broker that chooses targets randomly. Another possibility is, for example, the `FastestQueue` broker that chooses the target with the shortest estimated queue waiting time. For an overview of brokers, please refer to Section 2.2.1.

Example: `brokername=Data`

brokerarguments

This attribute is used in case a broker comes with arguments. This corresponds to the parameter that follows column in the `-b` command line option.

Example: `brokerarguments=cow`

joblist

The file storing information about active jobs. This file will be used by commands such as `arcsub`, `arcstat`, `arcsync` etc. to read and write information about jobs. This attribute corresponds to the `-j` command line option. The default location of the file on Linux platforms is in the $HOME/.arc directory with the name `jobs.dat`.

Example:

```
joblist=/home/user/run/jobs.dat
joblist=C:\\run\jobs.dat
```

joblisttype

The format to use when creating a new job list file for storing information about active jobs. Two possible formats can be used, `XML` and `BDB`. When using the former, job information is stored in plain text as XML, while the latter will use a Berkeley database to store job information. There is no command line option corresponding to this attribute. The default format is `BDB`.

Note that this option is only used by tools writing job information such as `arcsub`, `arcsync`, `arctest`, etc. and only if they are instructed to use a non-existing/new job list file, i.e. create a new.

Example:

```
joblisttype=XML
joblisttype=BDB
```

bartender

Specifies default *Bartender* services. Multiple Bartender URLs should be separated with a blank space. These URLs are used by the `chelonia` command line tool, the Chelonia FUSE plugin and by the data tool commands `arccp`, `arcls`, `arcrm`, etc..

Example: `bartender=http://my.bar.com/tender`

proxypath

Specifies a non-standard location of proxy certificate. It is used by `arcproxy` or similar tools during proxy generation, and all other tools during establishing of a secure connection. This attribute corresponds to the `-P` command line option of `arcproxy`.

Example: `proxypath=/tmp/my-proxy`

keypath

Specifies a non-standard location of user's private key. It is used by `arcproxy` or similar tools during proxy generation. This attribute corresponds to the `-K` command line option of `arcproxy`.

Example: `keypath=/home/username/key.pem`

certificatepath

Specifies a non-standard location of user's public certificate. It is used by `arcproxy` or similar tools during proxy generation. This attribute corresponds to the `-C` command line option of `arcproxy`.

Example: `certificatepath=/home/username/cert.pem`

cacertificatesdirectory

Specifies non-standard location of the directory containing CA-certificates. This attribute corresponds to the `-T` command line option of `arcproxy`.

Example: `cacertificatesdirectory=/home/user/cacertificates`

cacertificatepath

Specifies an explicit path to the certificate of the CA that issued user's credentials.

Example: `cacertificatepath=/home/user/myCA.0`

vomsserverpath

Specifies non-standard path to the file which contians list of VOMS services and associated configuration parameters. This attribute corresponds to the `-V` command line option of `arcproxy`.

Example: `vomsserverpath=/etc/voms/vomses`

jobdownloaddirectory

Sets directory which will be used as the default job download directory. This attribute corresponds to the `-D` command line option of `arcget`.

Example: `jobdownloaddirectory=/home/myjobs`

4.2 Service blocks

These blocks are only available in ARC 12.05 and above!

Each service is configured through its own block. Each service has to have unique alias name, which is used to refer to this service. The services can be grouped into multiple groups. Then the name of the group can be used in the command line to select all members of the group. Possible names of blocks are `[registry/<alias>]` for registry services and `[computing/<alias>]` for computing services.

Example:

```
[registry/index1]
url = ldap://index1.nordugrid.org:2135/Mds-Vo-name=NorduGrid,o=grid
registryinterface = org.nordugrid.ldapegiis
default = yes
group = favs

[registry/index2]
url = ldap://index2.nordugrid.org:2135/Mds-Vo-name=NorduGrid,o=grid

[computing/myce]
url = myce.example.com

[computing/ce]
url = https://ce.example.com:60000/arex
infointerface = org.ogf.glue.emies.resourceinfo
submissioninterface = org.ogf.glue.emies.activitycreation
default = yes
group = favs
```

url

The URL of the service. The URL can be shortened to the domain name, in which case the client will try to guess the missing parts (protocol, port, paths). This is the only **mandatory** option in a service block.

Example: `url=https://example.com:60000/arex`

default

Setting this to `yes` indicates that this service should be used by default by the commands if there were no computing elements or registries given as command line arguments. If default is not set then the service can only be enabled through command line options.

Example: `default=yes`

group

The name of the group the service belongs to. Services can be selected by specifying the name of the group as command line arguments instead of the alias of the service. A service can belong to multiple groups.

Example:

```
group=favs
group=fast
```

infointerface

The interface of the service through which the computing element information should be retrieved Example:

```
infointerface=org.nordugrid.ldapng
```

Possible information interfaces are:

- org.nordugrid.ldapng
- org.nordugrid.ldapglue1
- org.nordugrid.ldapglue2
- org.nordugrid.wsrfglue2
- org.ogf.glue.emies.resourceinfo

submissioninterface

The interface of the service with which the jobs should be submitted.

Example:

```
submissioninterface=org.nordugrid.gridftpjob
```

Possible submission interfaces are:

- org.nordugrid.gridftpjob
- org.ogf.bes
- org.ogf.glue.emies.activitycreation

registryinterface

The interface of the service with which the registry can be queried.

Example:

```
registryinterface=org.nordugrid.ldapegiis
```

Possible registry interfaces are:

- org.nordugrid.ldapegiis
- org.nordugrid.emir (*subject to change*)

4.3 srms.conf

If any data management commands are used with the Storage Resource Management (SRM) [6] protocol, the file

$HOME/.arc/srms.conf

(or its analogue on non-Linux platforms) may be created to store cached information on these services. For more information see the description inside this file.

4.4 Block [alias]

This block is deprecated in ARC 12.05!

Users often prefer to submit jobs to a specific site; since contact URLs (and especially end-point references) are very long, it is very convenient to replace them with aliases. Block [alias] simply contains a list of alias-value pairs.

Alias substitutions is performed in connection with the -c command line switch of the ARC clients.

Aliases can refer to a list of services (separated by a blank space).

Alias definitions can be recursive. Any alias defined in a list that is read before a given list can be used in alias definitions in that list. An alias defined in a list can also be used in alias definitions later in the same list.

Examples:

```
[alias]

arc0=computing:ARC0:ldap://ce.ng.org:2135/nordugrid-cluster-name=ce.ng.org,
Mds-Vo-name=local,o=grid
arc1=computing:ARC1:https://arex.ng.org:60000/arex
cream=computing:CREAM:ldap://cream.glite.org:2170/o=grid
crossbrokering=arc0 arc1 cream
```

4.5 Deprecated configuration files

ARC configuration file in releases 0.6 and 0.8 has the same name and the same format. Only one attribute is preserved (timeout); other attributes unknown to newer ARC versions are ignored.

In ARC $\leq$ 0.5.48, configuration on Linux platforms was done via files $HOME/.ngrc, $HOME/.nggiislist and $HOME/.ngalias.

The main configuration file $HOME/.ngrc could contain user's default settings for the verbosity level, the information system query timeout and the download directory used by ngget. A sample file could be the following:

```
# Sample .ngrc file
# Comments starts with #
NGDEBUG=1
NGTIMEOUT=60
NGDOWNLOAD=/tmp
```

If the environment variables NGDEBUG, NGTIMEOUT or NGDOWNLOAD were defined, these took precedence over the values defined in this configuration. Any command line options override the defaults.

The file `$HOME/.nggiislist` was used to keep the list of default GIIS server URLs, one line per GIIS (see `giis` attribute description above).

The file `$HOME/.ngalias` was used to keep the list of site aliases, one line per alias (see `alias` attribute description above).

Acknowledgements

This work was supported in parts by: the Nordunet 2 program, the Nordic DataGrid Facility, the EU KnowARC project (Contract nr. 032691), the EU EMI project (Grant agreement nr. 261611) and the Swedish Research council via the eSSENCE strategic research program.

Bibliography

[1] A. Anjomshoaa et al. Job Submission Description Language (JSDL) Specification, Version 1.0 (first errata update). GFD-R.136, July 2008. URL `http://www.gridforum.org/documents/GFD.136.pdf`.

[2] M. Ellert, B. Mohn, I. Márton, and G. Rőczei. *libarcclient – A Client Library for ARC*. The NorduGrid Collaboration. URL `http://www.nordugrid.org/documents/client_technical.pdf`. NORDUGRID-TECH-20.

[3] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997. Available at: http://www.globus.org.

[4] A. Konstantinov. *The ARC Computational Job Management Module – A-REX*. The NorduGrid Collaboration. URL `http://www.nordugrid.org/documents/a-rex.pdf`. NORDUGRID-TECH-14.

[5] F. Pacini and A. Maraschini. *Job Description Language attributes specification*, 2007. URL `https://edms.cern.ch/document/590869/1`. EGEE-JRA1-TEC-590869-JDL-Attributes-v0-8.

[6] A. Sim, A. Shoshani, et al. The Storage Resource Manager Interface (SRM) Specification v2.2. GFD-R-P.129, May 2008. URL `http://www.ggf.org/documents/GFD.129.pdf`.

[7] O. Smirnova. *Extended Resource Specification Language*. The NorduGrid Collaboration. URL `http://www.nordugrid.org/documents/xrsl.pdf`. NORDUGRID-MANUAL-4.

[8] M. Smith and T. A. Howes. *LDAP : Programming Directory-Enabled Applications with Lightweigt Directory Access Protocol*. Macmillan, 1997.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the definition; numbers in roman refer to the pages where the entry is used.

A

arccat 17
arcclean 23
arccp 29
arcget 18
arcinfo 20
arckill 21
arcls 28
arcmkdir 31
arcproxy 7
arcrename 31
arcrenew 24
arcresub 26
arcresume 25
arcrm 30
arcsub 11
arcsync 20
arctest 32

B

broker 14

C

commands
  arccat 17
  arcclean 23
  arccp 29
  arcget 18
  arcinfo 20
  arckill 21
  arcls 28
  arcmkdir 31
  arcproxy 7
  arcrename 31
  arcrenew 24
  arcresub 26
  arcresume 25
  arcrm 30
  arcsub 11
  arcsync 20
  arctest 32
configuration
  bartender 42
  brokerarguments 42
  brokername 41
  cacertificatepath 43
  cacertificatesdirectory . . . 43
  certificatepath 43
  default 44
  defaultservices 40
  deprecated files 46
  group 44
  infointerface 41, 45
  jobdownloaddirectory . . . 43
  joblist 42
  joblisttype 42

keypath 43
proxypath 42
registryinterface 45
rejectdiscovery 40
rejectmanagement 41
rejectservices 40
srms.conf 45
submissioninterface . . . 41, 45
timeout 41
url 44
verbosity 41
vomsserverpath 43

D

data management 28

J

job ID 13
job management 11

S

security 7
submit job 11

U

URL 35
options 36
URLs 35